Zoea

Research Note 13

What Makes Learning to Code So Hard for AI?

Edward McDaid & Sarah McDaid
14 Jun 2021

People have managed to build AI systems that do lots of interesting and useful things. Yet building an AI that can code has proved elusive - until recently. Zoea is the first and so far the only AI that enables people to produce software of any size. So why has it taken so long for AI to learn to code?

Most people tend to get a distorted view of artificial intelligence. On one hand we have the all-knowing oracles from science fiction that can solve any problem - usually couched as a pithy question. At the other extreme we have the AI systems that we have actually managed to build so far - which tend to do one very specific thing such as recognising faces or turning text into speech. In themselves, contemporary AI systems represent significant achievements yet there is something very obviously missing from both the real world and sci-fi incarnations of AI. That element is collaboration.

People and AI can and do already participate in some of the same workflows. For example the approval process for a bank loan will often include both algorithmic and human steps. However, this isn't really a case of a human and an AI working together to solve a problem. All of the pertinent information is available at the outset and the flow of information tends to be in one direction. In situations like this both the human and the AI are effectively following defined processes and business rules. In effect they're both executing predefined algorithms. Indeed the term algorithm has come to be a widely misused and somewhat ironic synonym for AI.

Real collaboration involves more than simply processing information and many of the activities that benefit from collaboration are also creative in one way or another. Collaborating parties tend to play relatively symmetric and complementary roles, and communication is invariably bidirectional. This leads to processes that are iterative.

Software development is a prime example of an activity that benefits from collaboration. It is also something that has proved notoriously difficult to teach an AI. This is because developing software is more than just coding. Virtually all software is created with reference to some form of requirement. However, software requirements are rarely complete, correct and stable - they also tend to change and grow over time. As a result a significant part of teaching an AI to code is getting the AI to understand the requirements.

Building an AI that can understand requirements would be somewhat easier if the relationship between the requirements and the program was always obvious. Unfortunately this is rarely the case. Even fairly simple software can quickly transform and mangle data out of all recognition. Also, requirements don't always spell out every detail, they can also be ambiguous, inconsistent or just plain wrong. This means that in order to understand requirements an AI needs to create potentially many different hypotheses about the code being specified. To do this the AI must also be able to execute those hypotheses to evaluate both the internal state of the data and the end result.

Unfortunately there is no way to determine whether a given program produces a particular output without executing it - otherwise it would be possible to solve the halting problem which as Alan Turing famously proved is impossible. This means that there is simply no way to create an AI that somehow directly maps the requirements to the corresponding code in a single step. The implication is that trying to build an AI that can code by simply training it with lots of examples of requirements and programs is - to say the least - highly problematic. It is of course still possible to build such an AI but not through training alone.

Learning to code with any conventional programming language is not particularly easy. Nevertheless software development is a well understood formal system with a finite number of rules. Regardless of how we choose to build our AI the end product will contain some sort of embodiment of those rules. With Zoea we decided to make those rules explicit. This is partly because we know what they are but also because we can build and test them separately, and also know when we're done. Equally importantly we can

understand where a particular hypothesis has come from and why it was chosen over another. Explanation and transparency in AI aren't just niceties - in some cases they are prerequisites in order to build an AI in the first place.

Software development is an iterative process that - at least for the foreseeable future - will continue to include human stakeholders. Building an AI that is capable of collaboration requires an understanding of the respective roles that the AI and people will play, and how they will communicate. This includes communication about the process and reasoning as well as the problem domain. In the case of Zoea definition of the protocol was a key factor in determining the overall feasibility of the approach. It also incidentally led to a radically new software development paradigm.

Collaboration is based on shared understanding which in turn requires agreed meaning. For an AI that can code, such as Zoea, shared understanding and agreed meaning are provided by executable models of software. Developing collaborative AI in other domains will present similar challenges and require equivalent models. AI developers will quickly find they need to adopt a broader range of approaches if they are to deliver systems of this kind.

Learn more at **zoea.co.uk**